AD NUMBER
AD823871
LIMITATION CHANGES
TO: **Approved for public release; distribution is unlimited. Document partially illegible.**
FROM: **Distribution authorized to U.S. Gov't. agencies and their contractors; Critical Technology; OCT 1967. Other requests shall be referred to Air Force Technical Application Center, VELA Seismological Center, Washington, DC. Document partially illegible. This document contains export-controlled technical data.**
AUTHORITY
aftac ltr, 11 feb 1960

FINITE FOURIER TRANSFORM THEORY AND ITS APPLICATION TO THE COMPUTATION OF CONVOLUTIONS, CORRELATIONS, AND SPECTRA

17 October 1967

Prepared For

AIR FORCE TECHNICAL APPLICATIONS CENTER
Washington, D. C.

By
Douglas W. McCowan
TELEDYNE, INC.

Under

Project VELA UNIFORM

Sponsored By

ADVANCED RESEARCH PROJECTS AGENCY
Nuclear Test Detection Office
ARPA Order No. 624

FINITE FOURIER TRANSFORM THEORY AND ITS APPLICATION TO THE COMPUTATION OF CONVOLUTIONS, CORRELATIONS, AND SPECTRA

SEISMIC DATA LABORATORY REPORT NO. 168 (REVISED)

AFTAC Project No.:	VELA T/6702
Project Title:	Seismic Data Laboratory
ARPA Order No.:	624
ARPA Program Code No.:	5810
Name of Contractor:	TELEDYNE, INC.
Contract No.:	F 33657-67-C-1313
Date of Contract:	2 March 1967
Amount of Contract:	$ 1,736,617
Contract Expiration Date:	1 March 1968
Project Manager:	William C. Dean (703) 836-7644

P. O. Box 334, Alexandria, Virginia

AVAILABILITY

This document is subject to special export controls and each transmittal to foreign governments or foreign national may be made only with prior approval of Chief, AFTAC.

This research was supported by the Advanced Research Projects Agency, Nuclear Test Detection Office, under Project VELA-UNIFORM and accomplished under the technical direction of the Air Force Technical Applications Center under Contract F 33657-67-C-1313.

TABLE OF CONTENTS

ABSTRACT

The theory of finite Fourier transforms is developed from the definitions of infinite transforms and applied to the computation of convolutions, correlations, and power spectra. Detailed procedures for these computations are given, including listings and writeups of FORTRAN subroutines.

1. INTRODUCTION

For the past several months, E. A. Flinn, J. F. Claerbout, and I have been examining some practical and computational aspects of the theory of Fourier transforms. These efforts have resulted in a set of programs for performing operations on time series based on the Cooley-Tukey (References 1,2) hyper-rapid Fourier transform method. Using this method, computations on seismic array data such as the calculation of convolutions, correlations, spectra, and digital filters have been speeded up by factors of three or four and sometimes even ten. The purpose of this report is to communicate these results in a straight-forward manner and to offer some motivation for their derivation as well as for future efforts in this area. Writeups and listings of the programs discussed here are included as appendices to this report.

2. THE FINITE AND DISCRETE FOURIER TRANSFORMS

In the case of continuous data of infinite length, the Fourier transform pair is usually written as:

$$A(\omega) = (2\pi)^{-\frac{1}{2}} \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

$$f(t) = (2\pi)^{-\frac{1}{2}} \int_{-\infty}^{\infty} A(\omega)e^{i\omega t}d\omega \qquad (1)$$

The first of these, going from time to frequency, is referred

to as the direct transform and the other as the inverse transform. Sometimes the direct transform is written with a factor of 1 in front of the integral and the inverse with a factor of $1/2\pi$. These are, of course, equivalent to the above definition. Quantities of interest, such as spectra, etc., involve magnitudes or squares of one transform and the factor must be inserted or taken out, depending on which definition is used, to preserve the proper dimensions.

Two drawbacks of these definitions for digital computations are apparent: First, the integrals must be approximated by sums in the digital computer, which implies that both transforms involve sampled variables. Second, the infinite limits on the sums are impossible in practice. Clearly, these sums must be truncated, as they do not in general converge over a finite interval. As a result Fourier transforms as such are never really computed by a digital computer. Instead, the complex samples of a direct transform are approximated by the cosine and sine coefficients of Fourier series representation of the input data. The definitions for these are:

if $$x(t) = \sum_{n=0}^{\infty} \left[a_n \cos(\pi n t/T) + b_n \sin(\pi n t/T)\right], \tag{2}$$

then $$a_o = \frac{1}{T}\int_o^T x(t)\,dt \qquad b_o = 0$$

$$a_n = \frac{2}{T}\int_o^T x(t)\cos(\pi n t/T)\,dt \tag{3}$$

$$b_n = \frac{2}{T} \int_o^T x(t) \sin (\pi nt/T)\, dt$$

If N samples of the data are taken at equally spaced intervals $\Delta t = T/N$, the integrals (3) becomes sums and the frequency sum in (2) goes from DC to the folding frequency, i.e., $k = 0$ to $N/2T$. The equations are then written as:

$$x(j) = \sum_{k=0}^{N/2} \left[a_k \cos (2\pi jk/N) + b_k \sin (2\pi jk/N) \right] \tag{4}$$

$$a_o = \frac{1}{N} \sum_{j=0}^{N-1} x(j) \qquad b_o = 0$$

$$a_k = \frac{2}{N} \sum_{j=0}^{N-1} x(j) \cos (2\pi jk/N) \qquad b_k = \frac{2}{N} \sum_{j=0}^{N-1} x(j) \sin (2\pi jk/N), \tag{5}$$

where t has been replaced by $j\Delta t$. By now defining:

$$A(k) = \frac{1}{2} (a_k - i\, b_k), \quad A(0) = a_o \tag{6}$$

and realizing that a real time series contains only real points, we can write (4) as:

$$x(j) = \sum_{k=0}^{N/2} A(k) \exp (2\pi ijk/N) \tag{7}$$

A great deal of symmetry between the two transforms can be

preserved if the sum in (7) is summed up to N-1 . Redundant points in the spectrum are included (since the transforms are periodic) but the computational procedures are simplified. It is also convenient to split the factor of 1/N appearing in (5) into two factors of $1/\sqrt{N}$, one in front of each transform. By defining a complex number:

$$w = \exp\ (2\pi i/N) \quad , \tag{8}$$

the two transforms can now be written as

$$A(k) = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f(j)\ w^{-jk} \tag{9}$$

$$f(j) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} A(k)\ w^{jk} \tag{10}$$

It can be shown that the set of direct Fourier transform points, between DC and the folding frequency, contains the same amount of information as the real data series: The transform includes N/2 distinct points, which with the DC term makes a total of N/2 + 1 complex points. Equation (9) shows that both the DC and the folding frequency point are purely real; thus, the Fourier transform contains (N/2-1)*2+2*1 numbers. This is exactly the same amount of information contained in the real time series. It also suggests that the existence of one transform should imply the existence of the other.

If there are N/2+1 non-redundant points in the direct transform, then the sampling interval in frequency must be

$(N/2T)/(N/2) = 1/T$. Thus, the product of the time and frequency variables is:

$$i\omega t = i(2\pi k \cdot \frac{1}{T})\ (j\frac{T}{N}) = \frac{2\pi i}{N} jk \tag{11}$$

This equation relates the arguments in the two exponentials, one in the continuous transform and the other in the finite transform (Equations 1, 9, and 10).

3. TWO- AND THREE-DIMENSIONAL FOURIER TRANSFORMS

Two- and three-dimensional direct Fourier transforms are seen to be

$$A(k_1,k_2) = \frac{1}{\sqrt{N_1 N_2}} \sum_{j_1=0}^{N_1-1} \sum_{j_2=0}^{N_2-1} x(j_1,j_2)\ w_1^{-j_1 k_1}\ w_2^{-j_2 k_2} \tag{12}$$

and

$$A(k_1,k_2,k_3) = \frac{1}{\sqrt{N_1 N_2 N_3}} \sum_{j_1=0}^{N_1-1} \sum_{j_2=0}^{N_2-1} \sum_{j_3=0}^{N_3-1} x(j_1,j_2,j_3)\ w_1^{-j_1 k_1} \cdot w_2^{-j_2 k_2}\ w_3^{-j_3 k_3} \tag{13}$$

We can break up Equation (12) as follows:

$$A(k_1,k_2) = \frac{1}{\sqrt{N_2}} \sum_{j_2=0}^{N_2-1} B(k_1,j_2)\ w_2^{-j_2 k_2} \tag{14}$$

This calculation requires N_1 one-dimensional transforms; we have defined

$$B(k_1, j_2) = \frac{1}{\sqrt{N_1}} \sum_{j_1=0}^{N_1-1} x(j_1, j_2)\, w_1^{-j_1 k_1} \tag{15}$$

which requires N_2 one-dimensional transforms. Thus, $N_1 + N_2$ one-dimensional transforms are required to compute the single two-dimensional transform.

We can break up Equation (13) as follows:

$$A(k_1, k_2, k_3) = \frac{1}{\sqrt{N_3}} \sum_{j_3=0}^{N_3-1} C(k_1, k_2, j_3)\, w_3^{-j_3 k_3} \tag{16}$$

which requires $N_1 N_2$ one-dimensional transforms; we have defined

$$C(k_1, k_2, j_3) = \frac{1}{\sqrt{N_1 N_2}} \sum_{j_1=0}^{N_1-1} \sum_{j_2=0}^{N_2-1} x(j_1, j_2, j_3)\, w_1^{-j_1 k_1}\, w_2^{-j_2 k_2} \tag{17}$$

which requires N_3 two-dimensional transforms. Thus, $N_1 N_2$ one-dimensional transforms and N_3 two-dimensional transforms are needed to compute the single three-dimensional transform.

4. ALGEBRAIC DISCUSSION

Equations (9) and (10) suggest a more elegant and compact way to write the two transforms. We define the vector $\underset{\sim}{A}$ as the transform with elements $(\underset{\sim}{A})_k = A(k)$, and define the vector $\underset{\sim}{F}$ as the time series with elements $(\underset{\sim}{F})_j = F(j)$. The process of transforming is seen to be equivalent to matrix multiplication by a matrix W whose elements are $(W)_{jk} = w^{jk}$

$$\underset{\sim}{A} = W^{+}\underset{\sim}{f} \tag{18}$$

and $$\underset{\sim}{f} = W\underset{\sim}{A} \ , \tag{19}$$

where the dagger indicates Hermitian conjugation. Substituting (19) into (18) gives the following important identity:

$$WW^{+} = W^{+}W = I \quad . \tag{20}$$

This is the definition of unitarity for the transformation W It is a generalization of orthogonality for complex matrices and assures Parseval's theorem:

$$\underset{\sim}{A}^{+}\underset{\sim}{A} = \underset{\sim}{f}^{+}\underset{\sim}{f} \quad . \tag{21}$$

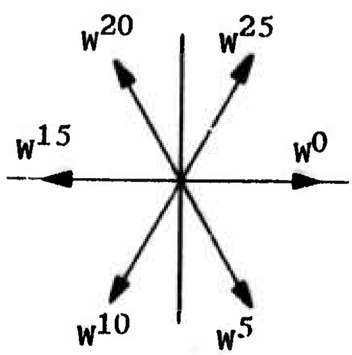
W preserves "length" between the two domains. The identity is actually proved by writing out the terms in the product:

$$\frac{1}{N}\sum_{m=0}^{N-1}\left[\exp(2\pi i/N)\right]^{jm}\left[\exp(-2\pi i/N)\right]^{mk} = \delta_k^j$$

or

$$\frac{1}{N} \sum_{m=0}^{N-1} W^{m(j-k)} = \delta_k^j \tag{22}$$

This last important relation is seen to be true by the use of a phase diagram:



for $j - k = 5$ and $N = 6$

The Cooley-Tukey method factors the W matrix, if its order is a power of two, into L + 1 sparse matrices, where L is the power of two:

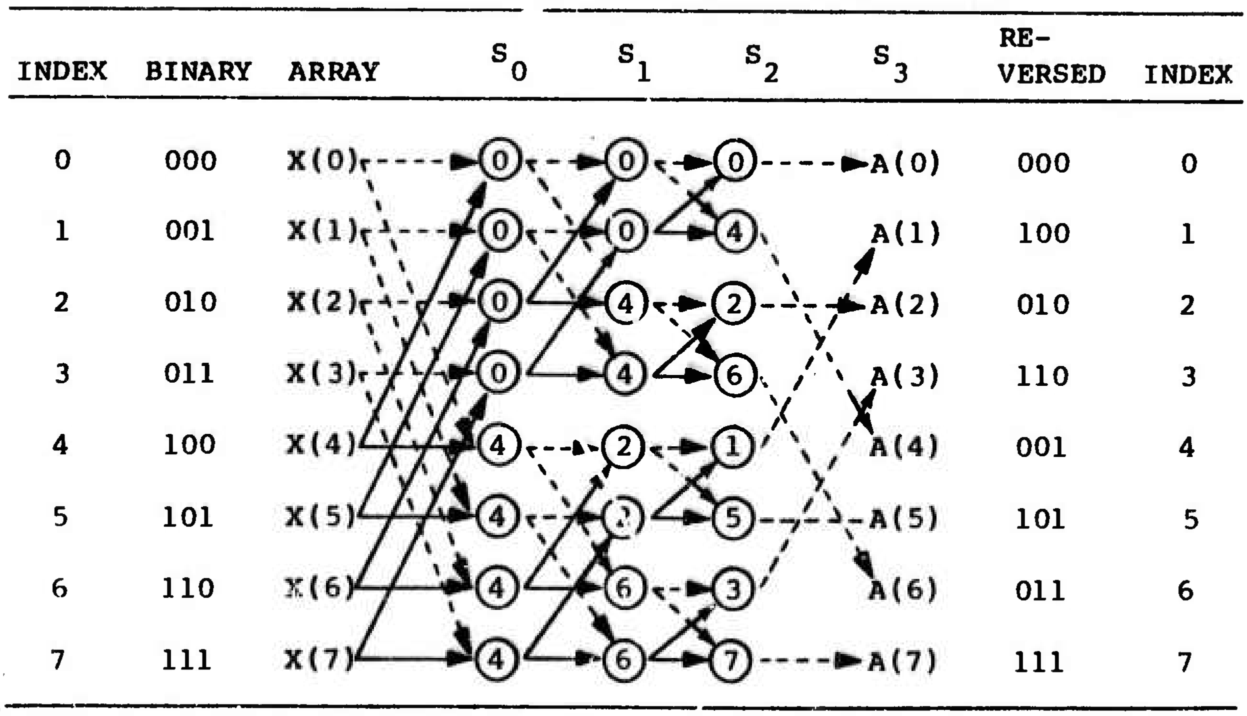
$$W = S_L \, S_{L-1} \ldots S_1 \, S_0$$

Multiplying L + 1 times by these sparse matrices can in some cases reduce the computing time by many tens of times. This factorization is proved by Good (3) and organized for computation by Rader (4).

For the case N = 8 the W matrix is:

$$W = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w^{1} & w^{2} & w^{3} & w^{4} & w^{5} & w^{6} & w^{7} \\ 1 & w^{2} & w^{4} & w^{6} & w^{8} & w^{10} & w^{12} & w^{14} \\ 1 & w^{3} & w^{6} & w^{9} & w^{12} & w^{15} & w^{18} & w^{21} \\ 1 & w^{4} & w^{8} & w^{12} & w^{16} & w^{20} & w^{24} & w^{28} \\ 1 & w^{5} & w^{10} & w^{15} & w^{20} & w^{25} & w^{30} & w^{35} \\ 1 & w^{6} & w^{12} & w^{18} & w^{24} & w^{30} & w^{36} & w^{42} \\ 1 & w^{7} & w^{14} & w^{21} & w^{28} & w^{35} & w^{42} & w^{49} \end{pmatrix} \tag{22.1}$$

The L + 1 = 4 transformations are graphically illustrated by the following diagram in Rader's notation (Reference 4):



Each solid line represents a multiplication by w to the

power indicated in the circle, and each dotted line represents a simple addition into that element of the array. No additional storage is used by this process. The results of each transformation are stored on top of the original data, and the last transformation, which is a simple interchange, gives the desired Fourier transform. Note also that the succession of numbers in the circles is the bit-reversed representation of the sequence of indices in order. They can be stored in a table or generated successively by a reverse-add procedure. For reasons of space and simplicity, we have chosen the latter route.

The S matrices are:

$$S_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 & w^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & w^0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & w^0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & w^0 \\ 1 & 0 & 0 & 0 & w^4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & w^4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & w^4 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & w^4 \end{pmatrix}$$

$$S_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & w^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & w^0 & 0 & 0 & 0 & 0 \\ 1 & 0 & w^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & w^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & w^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & w^2 \\ 0 & 0 & 0 & 0 & 1 & 0 & w^6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & w^6 \end{pmatrix}$$

$$S_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & w^0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & w^4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & w^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & w^6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & w^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & w^5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & w^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & w^7 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

5. HIGH-SPEED CORRELATIONS AND CONVOLUTIONS

By computing Fourier transforms with this finite Fourier series-like method an important condition is put on the time series. As in regular Fourier series the input is assumed to be periodic with period T and the integrals or sums are computed over a single period. There is also the effect of cutting off the spectrum at the folding frequency. Sines and cosines of finite wavelength will repeat again outside the region of interest. This fact in itself is not bothersome but becomes a serious complication in the computation of convolutions and correlations. Convolutions and correlations as usually computed assume the time series to be zero outside the region of interest. Therefore, the integrals or sums in computing them are summed out only over the non-zero terms. When multiplying together two finite Fourier transforms (or the complex conjugate of one times the other) the periodicity of the time series means that elements which have been shifted past the end of a period reappear at the beginning. This process is called circular convolution or correlation and its effects are unavoidable when straightforwardly computing lagged products with finite Fourier transforms. This is illustrated below:

$$X_1 = (3,\ 0,\ -1,\ 2)$$

$$X_2 = (\ -2,\ 2,\ -1,\ 3\ )$$

$$R^c_{12} = (1,\ -1,\ 3,\ 5) \quad \text{for 100\% positive lags;}$$

$$= (1,\ 5,\ 3,\ -1) \quad \text{for 100\% negative lags.}$$

Circular convolution is therefore written:

$$R_{ij}^{C}(t) = \sum_{\tau=0}^{T-1} x_i(\tau)\, x_j(t + \tau) \tag{23}$$

where $x_m(t + T) = x_m(t)$ for all m .

The proof that this kind of correlation is equal to the transform of the absolute product of the two finite transforms follows below:

$$\sum_{t=0}^{T-1} R_{ij}^{C}(t)\, w^{-tk} = \sum_{t=0}^{T-1} \sum_{\tau=0}^{T-1} x_i(\tau)\, x_j(t + \tau)\, w^{-tk} \tag{24}$$

$$= \sum_{\tau=0}^{T-1} \sum_{q=\tau}^{T-1+\tau} x_i(\tau)\, x_j(q)\, w^{-(q-\tau)k} \qquad q = t + \tau$$

$$= \sum_{\tau=0}^{T-1} x_i(\tau)\, w^{\tau k} \sum_{q=0}^{T-1} x_j(q)\, w^{-qk}$$

$$= A_i^{*}(k)\, A_j(k) .$$

On the other hand the transient correlation for positive lags is defined by the following:

$$R_{ij}^{T}(t) = \sum_{\tau=0}^{T-1-t} x_i(\tau)\, x_j(t + \tau) \tag{25}$$

where the upper limit on the sum simulates the desired zeros in the time series outside the region of interest. This is illustrated below:

$$X_1 = (3,\ 0,\ -1,\ 2)$$

$$X_2 = (-2,\ 2,\ -1,\ 3)$$

$$R^T_{12} = (1,\ 3,\ -3,\ 9) \quad \text{for 100\% positive lags;}$$

$$= (1,\ -4,\ 6,\ -4) \quad \text{for 100\% negative lags.}$$

The finite Fourier transform of this R^T is thus not the product of the two individual transforms. However, by filling zeros into the second half of each data series nd computing their transforms out to twice their actual length, a good estimate of the spectrum may be obtained. In addition, the negative lags in the correlation appear, thus giving a more mathematically satisfying result. This is illustrated below:

$$X_1 = (3,\ 0,\ -1,\ 2,\ 0,\ 0,\ 0,\ 0)$$

$$X_2 = (-2,\ 2,\ -1,\ 3,\ 0,\ 0,\ 0,\ 0)$$

$$R^C_{12} = (1,\ 3,\ -3,\ 9,\ 0,\ -4,\ 6,\ -4) \quad \text{for 100\% positive lags.}$$

The two modified transforms thus are:

$$F_i(k) = \sum_{t=0}^{2T-1} x_i(t)\ w^{-tk} \qquad x_i(t) = 0,\ T \le t \le 2T-1$$

$$S_{ij}(k) = F_i(k)^* \ F_j(k) = \sum_{t=0}^{2T-1} x_i(t)\ w^{tk} \sum_{\tau=0}^{2T-1} x_j(\tau)\ w^{-\tau k}$$

$$R^{?}_{ij}(s) = \sum_{k=0}^{2T-1} F_i(k)^* F_j(k)\, w^{ks} =$$

$$\sum_{t=0}^{2T-1} \sum_{\tau=0}^{2T-1} x_i(t)\, x_j(\tau) \sum_{k=0}^{2T-1} w^{k(t+s-\tau)} \quad . \tag{26}$$

Now from (22) the last sum becomes a Kronecker delta function and the other sum is collapsed to give:

$$R^{?}_{ij}(s) = \sum_{t=0}^{2T-1} x_i(t)\, x_j(t+s) = R^{T}_{ij}(s) \quad .$$

The last equality following from the original assumption that $x_i(t) = 0,\ T \le t \le 2T-1$. Transient correlations for 100% lags are therefore computed by forming the absolute product of two transforms, each computed out to twice the length of the original data series with zeros filled into the second halves.

Non-circular or transient convolutions are computed in much the same way, except that the transforms have to be computed out to a length equal to the sum of the lengths of the time series and the filter, with the appropriate number of zeros filled into each. The convolution theorem is proved in the same fashion.

$$A(k) = \sum_{\tau=0}^{T+S-1} a(\tau)\, w^{-\tau k} \qquad a(\tau) = 0\ , \quad S \le \tau \le T + S - 1$$

$$\overline{X}(k) = \sum_{t=0}^{T+S-1} x(t)\, w^{-tk} \qquad x(t) = 0 \qquad T \le t \le T + S - 1$$

$$\sum_{k=0}^{T+S-1} A(k)\, \overline{X}(k)\, w^{ku} = \sum_{\tau=0}^{T+S-1} \sum_{t=0}^{T+S-1} a(\tau)\, x(t) \sum_{k=0}^{T+S-1} w^{k(u-\tau-t)}$$

$$\sum_{k=0}^{T+S-1} A(k)\, x(k)\, w^{ku} = \sum_{\tau=0}^{T+S-1} a(t)\, x(u-t) = y(u) \tag{27}$$

Where y(u) is now the "filtered" output of the filter a acting on X. Convolutions are therefore computed by forming the product of the two transforms, each computed out to a length equal to their sum with zeros filled into the extra lengths. Detailed procedures for these computations are listed in Appendix A.

REFERENCES

1. Cooley, J. W. and Tukey, J. W., 1965, An Algorithm for the Machine Calculation of Complex Fourier Series: Math. of Comp., Vol. 19, pp. 297-301.

2. Cooley, J. W., 1964, Private Communication

3. Good, I. J., 1958, The Interaction Algorithm and Practical Fourier Series: J. Roy. Stat. Soc., Ser. B., Vol. 20, pp. 361-372; Addendum, Vol. 22, 1960, pp. 372-375.

4. Rader, C. M., 1965, Private Communication

5. Sande, G., 1965, Private Communication

6. Stockham, T. G., 1966, High-Speed Convolution and Correlation: AFIPS Conference Proceedings, Vol. 28, pp. 229-233.

APPENDIX A - PROCEDURES

FINITE FOURIER TRANSFORM THEORY AND ITS APPLICATION TO THE COMPUTATION OF CONVOLUTIONS, CORRELATIONS, AND SPECTRA

PROCEDURE FOR CALCULATING AN AUTO-SPECTRUM AND AN AUTO-CORRELATION

```
DIMENSION X(2*LX+2), CX(LX+1)
EQUIVALENCE (X,CX)
TYPE COMPLEX CX, CONJG
LX = 2**N
```

1) Erase 2*LX+2 points in X; the extra complex point is needed by COOLER to return the point at the folding frequency.

2) Read the data channel into X(1) through X(LX).

3) CALL COOLER(N+1,CX). The Fourier transform of X and the necessary zeros on the end of the data is now stored in CX, LX+1 complex points long, representing frequencies between DC and the folding frequency.

4) Go through the LX+1 complex points in CX, and:

   CX(I) = [CONJG(CX(I))*CX(I)]/LX

   that is,

   $Re[CX(I)] = (Re[CX(I)]^2 + Im[CX(I)]^2)/LX$

   $Im[CX(I)] = 0.0$

   The auto-spectrum is the real part of CX, purely real and LX+1 points in length.

5) To get the auto-correlation, fill in the other LX-1 complex points in CX as required by COOL for inverse transforms, and call COOL:

```
    DO 1 I = 1,LX-1
  1 CX(LX+1+I) = CX(LX-I+1)
    CALL COOL(N+1,CX,+1.0)
```

   The auto-correlation is in the real part of CX, purely real and 2*LX points in length.

NOTE: CX must be dimensioned 2*LX if the auto-correlation is to be computed.

PROCEDURE FOR CALCULATING A CROSS SPECTRUM AND A CROSS-CORRELATION

```
DIMENSION X(2*LX+2), CX(LX+1), Y(2*LX+2), CY(LX+1)
EQUIVALENCE (X,CX), (Y,CY)
TYPE COMPLEX CX,CY
LX = 2**N
```

1) Erase 2*LX+2 points in both X and Y.

2) Read channel 1 into X and channel 2 into Y.

3) CALL COOLER(N+1,X)
   CALL COOLER(N+1,Y)

4) Go through the LX+1 complex points and overlay CX (or CY) with:

   CX(I) = [CONJG(CX(I))*CY(I)]/LX

   that is,

   Re[CX(I)] = (Re[CX(I)]*Re(CY(I)]+Im[CX(I)]*Im[CY(I)])/LX

   Im[CX(I)] = (Re[CX(I)]*Im[CY(I)]-Im[CX(I)]*Re[CY(I)])/LX

   The cross-spectrum between channel 1 and channel 2 (which is the complex conjugate of the cross-spectrum between channel 2 and channel 1) is now in CX, LX+1 points in length. The co-spectrum is in the real part of CX and the quad-spectrum is in the imaginary part of CX.

5) To get the cross-correlation, fill in the other LX-1 points in CX and call COOL:

```
    DO 1 I = 1,LX-1
  1 CX(LX+I+1) = CONJG(CX(LX-I+1))
    CALL COOL(N+1,CX,+1.0)
```

   The cross-correlation is in the real part of CX, purely real and 2*LX points in length.

   NOTE: CX must be dimensioned 2*LX if the cross-correlation is to be calculated.

PROCEDURE FOR CALCULATING THE CONVOLUTION OF TWO SERIES

```
   DIMENSION X(L+2), CX(½L+1), F(L+2), CF(½L+1)
  EQUIVALENCE (X,CX), (F,CF)
  TYPE COMPLEX CX,CF,CONJG
  L = 2**N
```

L here is the next power of 2 larger than LX+LF,,the combined length of the data and the filter.

1) Erase L+2 points in X and F.

2) Read the data into X(1) through X(LX) and the filter impulse response into F(1) through F(LF).

3) CALL COOLER(N,CX)
   CALL COOLER(N,CF)

4) Go through the ½L+1 complex points in CX, and:

   CX(I) = [CX(I)*CF(I)]/LX

   that is,

   Re[CX(I)] = (Re[CX(I)]*Re[CF(I)]-Im[CX(I)]*Im[CF(I)])/LX

   Im[CX(I)] = (Re[CX(I)]*Im[CF(I)]+Re[CF(I)]*Im[CX(I)])/LX

   The Fourier transform of X convolved with F is now in CX.

5) Fill in the rest of the points in CX as needed by COOL, and transform back. Note again that if the actual convolution is desired instead of the Fourier transform, CX must be dimensioned L.

```
     DO 1 I = 1, ½L-1
   1 CX(½L+I+1) = CONJG[CX(½L -I+1)]
     CALL COOL(N,CX,-1.0)
```

The convolution of X with F is now in the real part of CX, purely real, and LX+LF-1 points in length.

APPENDIX B - PROGRAM LISTINGS

FINITE FOURIER TRANSFORM THEORY AND ITS APPLICATION TO THE COMPUTATION OF CONVOLUTIONS, CORRELATIONS, AND SPECTRA

```
      SUBROUTINE COOL(N,X,SIGNI)
C
C
C        HYPER-RAPID FOURIER TRANSFORM USING COOLEY-TUKEY ALGORITHM
C
C
C        SEISMIC DATA LABORATORY, ALEXANDRIA, VA.  PROGRAMMED
C        26 FEBRUARY 1966 BY J. F. CLAERBOUT (MIT), D. W. MCCOWAN,
C        E. A. FLINN, AND J. GIBSON (TELEDYNE)
C
C        X IS A COMPLEX ARRAY USED FOR THE DATA SERIES AND THE
C
C                                                  N
C        TRANSFORM - THE NUMBER OF ELEMENTS OF X IS L = 2 .
C        SIGN = -1.0 FOR DIRECT FOURIER TRANSFORM AND +1.0 FOR INVERSE
C        FOURIER TRANSFORM  (BUT SEE BELOW FOR ARRANGEMENT OF DATA FOR
C        INVERSE TRANSFORM).
C
C        FOR DIRECT TRANSFORM, ON INPUT THE REAL PART OF X CONTAINS THE
C        DATA SERIES AND THE IMAGINARY PART OF X IS ZERO.  ON RETURN,
C        THE FOURIER COSINE SERIES EXPANSION OF THE DATA IS IN THE REAL
C        PART OF X, AND THE FOURIER SINE SERIES EXPANSION IS IN THE
C
C                                             N-1
C        IMAGINARY PART OF X.  EACH CONTAINS ONLY 2    + 1 NONREDUNDANT
C        POINTS.  THE COSINE EXPANSION IS SYMMETRIC ABOUT POINT NUMBER
C
C         N-1
C        2    + 1 AND THE SINE TRANSFORM IS ANTISYMMETRIC ABOUT
C        THIS POINT.
C
C        FOR EXAMPLE - N = 3 AND DATA = (0.,1.,0.,0.,0.,0.,0.,0.),
C        THEN REAL PART OF X = (0.,1.,0.,0.,0.,0.,0.,0.) AND IMAGINARY
C        PART OF X = (0.,0.,0.,0.,0.,0.,0.,0.) ON INPUT.
C        ON RETURN, REAL PART OF X = (1.000,.7071,0.,-.7071,-1.000,
C        -.7071,0.,+.7071) AND IMAGINARY PART OF X = (0.,-.7071,
C        -1.000,-.7071,0.,.7071,1.000,.7071).  POINT NUMBER 1
C        CORRESPONDS TO ZERO FREQUENCY, POINT NUMBER 5 CORRESPONDS
C        TO PI, THE FOLDING FREQUENCY.
C
C        TO DO AN INVERSE TRANSFORM, THE COSINE AND SINE SERIES MUST BE
C
C                                       N-1
C        FOLDED OVER ABOUT POINT NUMBER  2    + 1 BEFORE CALLING
C        COOL WITH SIGN = +1.0.  SUBROUTINE FTPACK CAN BE USED TO DO
C        THIS FOR YOU, CONVERTING AMPLITUDE AND PHASE BACK TO
C        SINE AND COSINE IF NEED BE.
C
C                                  -N
C        THERE IS A SCALE FACTOR OF  2   WHICH COOL DOES NOT APPLY.
C                                                         ---
C
C        THE USER CAN APPLY THE SCALE FACTOR EITHER TO THE DIRECT OR TO
C
C                                                        -N/2
C        THE INVERSE TRANSFORM, OR APPLY A  SCALE FACTOR OF 2      TO
```

```
C         FOR EXAMPLE, GIVEN THE INPUT DATA AS ABOVE, THE TWO STATEMENTS
C         CALL COOL(3,X,-1.0)
C         CALL COOL(3,X,+1.0)
C         WOULD CHANGE REAL PART OF X TO (0.,8.,0.,0.,0.,0.,0.,0.) AND
C         IMAGINARY PART OF X TO (0.,0.,0.,0.,0.,0.,0.,0.).
C
C
C
      DIMENSION X(1),INT(16),G(2)
      TYPE COMPLEX X,Q,W,HOLD
      EQUIVALENCE (G,W)
C
C         INITIALIZE
C
      LX = 2**N
      PI2=6.283185306
      FLX = LX
      FLXPI2=SIGNI*PI2/FLX
      DO 10 I=1,N
   10 INT(I) = 2**(N-I)
C
C         LOOP OVER N LAYERS
C
      DO 40 LAYER = 1,N
      NBLOCK = 2**(LAYER-1)
      LBLOCK=LX/NBLOCK
      LBHALF = LBLOCK/2
C
C         START SERIES AND LOOP OVER BLOCKS IN EACH LAYER
C
      NW = 0
      DO 40 IBLOCK=1,NBLOCK
      LSTART = LBLOCK*(IBLOCK-1)
C
C         COMPUTE W = CEXP(2.*PI*NW*SIGNI/LX)
C
      ARG=FLOATF(NW)*FLXPI2
      G(1)  = COSF(ARG)
      G(2)  = SINF(ARG)
C
C         THIS CAN BE SPEEDED UP BY USING A TABLE OF COSINES
C
C
C         COMPUTE ELEMENTS FOR BOTH HALFS OF EACH BLOCK
C
      DO 20 I=1,LBHALF
      J = I+LSTART
      K = J+LBHALF
      Q = X(K)*W
      X(K) = X(J)-Q
      X(J) = X(J)+Q
   20 CONTINUE
C
C         BUMP UP SERIES BY TWO (NOT ONE)
C
      DO 32 I=2,N
```

```
      II = I
      LL=INT(I).AND.NW
C
C        THIS LOGICAL OPERATION IS A MASK TO DETECT A ONE IN
C        THE APPROPRIATE BIT POSITION OF NW.  THIS STATEMENT WILL NOT
C        WORK ON IBM FORTRAN SYSTEMS.
C
      IF(LL)31,31,30
   30 CONTINUE
      NW = NW-INT(I)
   32 CONTINUE
   31 CONTINUE
      NW = NW+INT(II)
   40 CONTINUE
C
C        START SERIES TO BEGIN FINAL REPLACEMENT
C
      NW = 0
      DO 50 K=1,LX
C
C        CHOOSE CORRECT INDEX AND SWITCH ELEMENTS IF NOT ALREADY
C        SWITCHED
C
      NW1=NW+1
      IF(NW1-K)55,55,60
   60 HOLD=X(NW1)
      X(NW1)=X(K)
      X(K) = HOLD
   55 CONTINUE
C
C        BUMP UP SERIES BY ONE
C
      DO 70 I=1,N
      II = I
      LL=INT(I).AND.NW
      IF(LL)80,80,70
   70 NW = NW-INT(I)
   80 NW = NW+INT(II)
   50 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE COOLCON(INT,IOT,L,F,X)
      DIMENSION F(1),X(2,1),LAB(127)
C
C     DIMENSION F(N,L),X(2,ITEST),LAB(127)
C
C        MULTICHANNEL CONVOLUTION ROUTINE FOR TAPED DATA
C
C        INT IS THE INPUT SUBSET TAPE OF DATA CHANNELS
C        IOT IS THE OUTPUT SUBSET TAPE OF DATA CHANNELS
C        L IS THE NUMBER OF FILTER POINTS FOR EACH CHANNEL
C        F IS THE FILTER MATRIX
C        X IS A WORKING ARRAY CONTAINING AT LEAST 2*ITEST POINTS
C        ITEST IS THE NEXT POWER OF TWO LARGER THAN LX+L
C
C        D.W.MCCOWAN JULY 1966
C
      REWIND INT
      REWIND IOT
      READ(INT)LAB
      N=LAB(2)
      LX=LAB(3)
      ISUM=LX+L
      LAB(3)=LX-(L-1)
      WRITE(IOT)LAB
      DO 1 IND=1,13
      ITEST=2**IND
      IF(ISUM-ITEST)2,2,1
    2 NCOOL=IND
      GO TO 3
    1 CONTINUE
      PRINT 1000,LX,L
 1000 FORMAT(59H1BAD NEWS, ERROR IN COOLCON, DATA PLUS FILTER TOO LONG L
     1X= ,I6,5H, L= ,I6)
      STOP
    3 CONTINUE
      ITO2=ITEST/2
      ITO2P2=ITO2+2
      DO 10 IN=1,N
      CALL ERASE(2*ITEST,X)
      READ(INT)(X(1,M),M=1,LX)
      DO 11 IL=1,L
   11 X(2,IL)=F(IN+(IL-1)*N)
      CALL COOL(NCOOL,X,-1,0)
      X(1,1)=X(1,1)*X(2,1)/ITEST
      X(2,1)=0.0
      DO 20 IL=2,ITO2
      SAVE=(X(1,ITEST-IL+2)*X(2,ITEST-IL+2)+X(1,IL)*X(2,IL))/(2*ITEST)
      X(2,IL)=(X(1,ITEST-IL+2)**2-X(2,ITEST-IL+2)**2-X(1,IL)**2+X(2,IL)*
     1*2)/(4*ITEST)
   20 X(1,IL)=SAVE
      X(1,ITO2+1)=X(1,ITO2+1)*X(2,ITO2+1)/ITEST
      X(2,ITO2+1)=0.0
      DO 30 IL=ITO2P2,ITEST
      X(1,IL)=X(1,ITEST-IL+2)
   30 X(2,IL)=-X(2,ITEST-IL+2)
      CALL COOL(NCOOL,X,+1,0)



   10 WRITE(IOT)(X(1,M),M=L,LX)
      END FILE IOT
      REWIND IOT
      REWIND INT
      RETURN
      END
```

```
      SUBROUTINE COOLER(N,X)
C
C     DIMENSION X(2*L+2)
C
C        FOURIER TRANSFORM OF A REAL DATA SERIES
C        N MUST BE LESS THAN OR EQUAL TO  14
C
      DIMENSION X(2,1)
      M = N-1
      L = 2**M
      CALL COOL(M,X,-1,0)
      F = 3.141592653/FLOATF(L)
      SAVE=X(1)
      X(1)=X(1)+X(2)
      X(1,L+1)=SAVE-X(2)
      X(2,1)=X(2,L+1)=0.0
      LL=L/2
      DO 10 I=2,LL
      J = L-I+2
      A1=0.5*(X(1,I)+X(1,J))
      A2=0.5*(X(2,I)-X(2,J))
      B1=0.5*(-X(1,I)+X(1,J))
      B2=0.5*(-X(2,I)-X(2,J))
      ZI = I-1
      F1 = F*ZI
      G1=COSF(F1)
      G2=-SINF(F1)
      SAVE=B1
      B1=B1*G1-B2*G2
      B2=B2*G1+SAVE*G2
      X(1,I)=A1-B2
      X(2,I)=A2+B1
      X(1,J)=A1+B2
   10 X(2,J)=-A2+B1
      X(2,LL+1)=-X(2,LL+1)
      RETURN
      END
```

```
      SUBROUTINE COOLHLBR(N,X)
C
C
C        THIS COMPUTES THE HILBERT TRANSFORM OF A DATA SERIES,
C        USING THE HYPER-RAPID FOURIER TRANSFORM ROUTINE COOL
C        THIS PROGRAM THANKS TO JON CLAERBOUT
C
C
C        INPUTS -
C        N = LOG (BASE 2) OF NUMBER OF DATA POINTS
C        REAL(X) = DATA SERIES TO BE TRANSFORMED
C        IMAG(X) = 0
C
C        OUTPUTS -
C        REAL(X) = X AGAIN
C        IMAG(X) = HILBERT TRANSFORM OF X
C
C        THIS CALLS COOL
C
      DIMENSION X(1)
      TYPE COMPLEX X
      CALL COOL(N,X,-1.0)
      M = 2**N
      M1 = M/2+2
      DO 1 I=M1,M
    1 X(I) = (0.,0.)
      X(1) = .5*X(1)
      X(M1-1) = .5*X(M1-1)
      CALL COOL(N,X,+1.0)
      RETURN
      END
```

```
      SUBROUTINE COOLTWO(N,X,SIGN,A,B)
C
C
C        THIS USES COOL TO COMPUTE THE FOURIER TRANSFORM OF TWO
C          TIME SERIES AT ONCE
C
C        INPUTS -
C        N     LOG (BASE 2) OF NUMBER OF DATA POINTS
C        X     A COMPLEX ARRAY OF DATA.  THE FIRST TIME SERIES IS STORED
C          IN THE REAL PART OF X, AND THE SECOND IS STORED IN THE
C          IMAGINARY PART OF X.  IN OTHER WORDS, THE TWO SERIES ARE
C          MULTIPLEXED IN THE ARRAY X.
C        SIGN    = -1.0 FOR DIRECT TRANSFORM.  THIS SUBROUTINE
C          HAS NOT BEEN CHECKED OUT FOR TWO INVERSE TRANSFORMS
C          AT ONCE.
C
C
C        OUTPUTS -
C        A     COMPLEX FOURIER TRANSFORM OF THE FIRST DATA SERIES,
C          I.E., THE ONE STORED IN THE REAL PART OF X
C        B     FOURIER TRANSFORM OF THE SECOND DATA SERIES, I.E., THE
C          ONE STORED IN THE IMAGINARY PART OF X.
C        BOTH TRANSFORMS ARE OF LENGTH 2**(N-1) + 1  (SEE COOL WRITEUP)
C
C
      DIMENSION X(1),A(1),B(1)
      TYPE COMPLEX X,A,B,CONJG
      CALL COOL(N,X,SIGN)
      A(1) =      .5*(X(1)+CONJG(X(1)))
      B(1) = (0.,-.5)*(X(1)-CONJG(X(1)))
      M=2**N
      DO 10 K=2,M
      A(K)=0.5*(X(K)+CONJG(X(M+2-K)))
   10 B(K)=(0.0,-0.5)*(X(K)-CONJG(X(M+2-K)))
      RETURN
      END
```

```
      SUBROUTINE COOLVOLV(LX,X,LF,F)
      DIMENSION F(1),X(2,1)
C        SINGLE-CHANNEL CONVOLUTION USING COOL
C
C        THIS TAKES FOURIER TRANSFORM OF DATA AND FILTER, MULTIPLIES
C        THEM TOGETHER, AND TRANSFORMS BACK,
C
C        INPUTS -
C
C          LX      LENGTH OF DATA
C          LF      LENGTH OF FILTER
C          F       FILTER COEFFICIENTS DIMENSIONED  F(LF) IN CALLING PGM
C          X       DATA, DIMENSIONED  X(N) IN CALLING PGM, WHERE
C             N IS THE SMALLEST NUMBER WHICH IS A POWER OF 2 EXCEEDING
C                (LF+LX)*2
C
C          THE SUBROUTINE RETURNS X CONVOLVED WITH F, OF LENGTH
C            LF+LX-1, STORED CLOSE-PACKED IN X.
C
C          23 SEPTEMBER 1966    DWMCC
C
C
C        CHECK LENGTH RESTRICTION
C
      NX=LF+LX
      DO 10 I=1,13
      N=2**I
      IF(NX-N) 20,20,10
 10   CONTINUE
C
C        ERROR RETURN - LENGTH OF FILTERED RECORD WOULD EXCEED LIMIT
C
      LF=-LF
      RETURN
C
 20   NCOOL =I
C
C        ERASE WORKING SPACE IN X
C
      CALL ERASE(2*N-LX,X(LX+1))
C
C        MULTIPLEX DATA AND FILTER IN X
C
      DO 30 I=1,LX
      J=LX-I+1
 30   X(1,J)    = X(J)
      DO 35 I=1,LX
 35   X(2,I) = 0.0
      DO 40 I=1,LF
 40   X(2,I) = F(I)
C
C        TRANSFORM AND FIDDLE
C
      FN=N
      CALL COOL(NCOOL,X,-1.0)
      X(1,1) = X(1,1)*X(2,1)/FN
```

```
      X(2,1) = 0.0
      N2=N/2
      DO 50 IL=2,N2
      T = (X(1,N-IL+2)*X(2,N-IL+2)+X(1,IL)*X(2,IL))/(2.*FN)
      X(2,IL) =(X(1,N-IL+2)**2-X(2,N-IL+2)**2-X(1,IL)**2+X(2,IL)**2)/
     1 (4.*FN)
   50 X(1,IL)=T
      X(1,N2+1)=X(1,N2+1)*X(2,N2+1)/FN
      X(2,N2+1)=0.0
      N22=N2+2
      DO 60 IL=N22,N
      X(1,IL)=X(1,N-IL+2)
   60 X(2,IL)= -X(2,N-IL+2)
C
C        TRANSFORM BACK
C
      CALL COOL(NCOOL,X,+1.0)
C
C        CLOSE-PACK FILTERED DATA IN X
C
      DO 70 I=1,NX
   70 X(I) = X(1,I)
C
C
      RETURN
      END
```

```
      SUBROUTINE FT2DCOOL(X,N,M,SIGNI)
      DIMENSION X(N,M)
      TYPE COMPLEX X
C
C        2 DIMENSION FOURIER TRANSFORM USING COOL
C
      NCOOL=LOGF(FLOATF(N))/LOGF(2.0)+1.0E-6
      MCOOL=LOGF(FLOATF(M))/LOGF(2.0)+1.0E-6
      SCALEN=1.0/SQRTF(FLOATF(N))
      SCALEM=1.0/SQRTF(FLOATF(M))
      DO 1 IM=1,M
      CALL COOL(NCOOL,X(1,IM),SIGNI)
      DO 1 IN=1,N
    1 X(IN,IM)=X(IN,IM)*SCALEN
      CALL MATRA63(X,N,M,X)
      DO 2 IN=1,N
      INDEX=1+(IN-1)*M
      CALL COOL(MCOOL,X(INDEX,1),SIGNI)
      CALL SCALE(SCALEM,M,X(INDEX,1))
    2 CONTINUE
      CALL MATRA63(X,M,N,X)
      RETURN
      END
```

```
      SUBROUTINE FT3DCOOL(X,N,M,L,SIGNI)
      DIMENSION X(N,M,L)
      TYPE COMPLEX X
C
C        3 DIMENSION FOURIER TRANSFORM USING COOL
C
      LCOOL=LOGF(FLOATF(L))/LOGF(2.0)+1.0E-6
      SCALEL=1.0/SQRTF(FLOATF(L))
      DO 1 IL=1,L
      CALL FT2DCOOL(X(1,1,IL),N,M,SIGNI)
    1 CONTINUE
      CALL MATRA63(X,N*M,L,X)
      DO 2 IN=1,N
      DO 2 IM=1,M
      INDEX=1+(IN-1)*L+(IM-1)*L*N
      CALL COOL(LCOOL,X(INDEX,1,1),SIGNI)
      CALL SCALE(SCALEL,L,X(INDEX,1,1))
    2 CONTINUE
      CALL MATRA63(X,L,N*M,X)
      RETURN
      END
```

```
      SUBROUTINE MATRA03(A,N,M,B)
      DIMENSION A(2,1),B(2,1)
C
C        MATRIX TRANSPOSE ON COMPLEX ARRAYS
C
      MASK1=0000000000000001B
      MASK2=7777777777777776B
      NM=N*M
      DO 10 I=1,NM
      B(1,I)=A(1,I).OR.MASK1
   10 B(2,I)=A(2,I)
      JF=0
      ASSIGN 30 TO KSWH
      DO 100 I=1,NM
      GO TO KSWH,(30,50)
   30 JF=JF+1
      LL=B(1,JF).AND.MASK1
      IF(LL)30,30,40
   40 JO=JF-1
      ASSIGN 50 TO KSWH
      TEMPB1=B(1,JF)
      TEMPB2=B(2,JF)
   50 J1=JO/N+XMODF(JO,N)*M+1
      TEMPA1=B(1,J1)
      TEMPA2=B(2,J1)
      B(1,J1)=TEMPB1.AND.MASK2
      B(2,J1)=TEMPB2
      TEMPB1=TEMPA1
      TEMPB2=TEMPA2
      JO=J1-1
      IF(J1-JF)60,60,100
   60 ASSIGN 30 TO KSWH
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SPECIMUM(IT,JT,KT,X,L,LF,S)
      DIMENSION X(2,1),S(2,1)
C
C     DIMENSION X(2,N,N,LF),X(2,2*LX+2),S(2,LF)
C
C
C         SPECTRAL MATRIX FOR TAPED DATA
C
C         DATA MUST BE A POWER OF TWO IN LENGTH AND ON TAPE IT IN SUBSET
C         FORMAT.  SPECTRAL MATRIX IS RETURNED AS A F63 COMPLEX MATRIX
C         IN X.
C
C         IT-INPUT SUBSET TAPE
C         JT-SCRATCH
C         KT-SCRATCH
C         X-WORKING ARRAY AND RETURNED SPECTRAL MATRIX
C         L-NUMBER OF TIMES TO SMOOTH
C         LF-RETURNED LENGTH OF SPECTRAL ESTIMATES
C         S-WORKING ARRAY
C
C         PROGRAM TOO COMPLICATED TO DESCRIBE...
C
      REWIND IT
      REWIND JT
      REWIND KT
      READ(IT)LOST,N,LX
      LX2=2*LX
      NCOOL=LOGF(FLOATF(LX))/LOGF(2.0)+1.0E-6
      NSQ=N*N
      LF=LX/2**L+1
      LX2P2=LX2+2
      LX4=4*LX
      LX2P2T2=2*LX2P2
      LXP1=LX+1
      LX2P3=LX2+3
      IDC=0
      LF2=2*LF
      WRITE(JT)LOST,N,LX2P2
      WRITE(KT)LOST,N,LX2P2
      DO 10 IN=1,N
      CALL ERASE(LX4,X)
      READ(IT)(X(1,M),M=1,LX)
      CALL COOL(NCOOL+1,X,-1.0)
      WRITE(JT)(X(M),M=1,LX2P2)
      WRITE(KT)(X(M),M=1,LX2P2)
   10 CONTINUE
      END FILE JT
      END FILE KT
      REWIND IT
      REWIND JT
      REWIND KT
      DO 1 IN=1,N
      IND=IN+1
      CALL SKIPREC(IN,KT)
      READ(KT)(X(M),M=1,LX2P2)
      CALL DOTEM(X,X,LXP1,X(LX2P3))
```

```
      CALL SMOOTH(X(LX2P3),LXP1,L)
      CALL DISC63(IDC,1,X(LX2P3),LF2)
      IDC=IDC+9
      DO 60 JN=IND,N
      READ(KT)(X(M),M=LX2P3,LX2P2T2)
      CALL DOTEM(X,X(LX2P3),LXP1,X(LX2P3))
      CALL SMOOTH(X(LX2P3),LXP1,L)
      CALL DISC63(IDC,1,X(LX2P3),LF2)
      IDC=IDC+9
   60 CONTINUE
      REWIND KT
      ISAVE=KT
      KT=JT
      JT=ISAVE
    1 CONTINUE
      IDC=0
      DO 25 IN=1,N
      IND=IN+1
      CALL DISC63(IDC,0,S,LF2)
      IDC=IDC+9
      INDEX=IN+(IN-1)*N
      DO 26 IL=1,LF
      X(1,INDEX)=S(1,IL)
      X(2,INDEX)=S(2,IL)
      INDEX=INDEX+NSQ
   26 CONTINUE
      DO 27 JN=IND,N
      CALL DISC63(IDC,0,S,LF2)
      IDC=IDC+9
      INDEX1=IN+(JN-1)*N
      INDEX2=JN+(IN-1)*N
      DO 28 IL=1,LF
      X(1,INDEX1)=S(1,IL)
      X(2,INDEX1)=S(2,IL)
      X(1,INDEX2)=S(1,IL)
      X(2,INDEX2)=-S(2,IL)
      INDEX1=INDEX1+NSQ
      INDEX2=INDEX2+NSQ
   28 CONTINUE
   27 CONTINUE
   25 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SMOOTH(X,LENGTH,L)
C
C          THIS HANNING ROUTINE THANKS TO J CLAERBOUT
C
      DIMENSION X(2,LENGTH)
      LF=LENGTH
      LFM1=LF-1
      DO 1 IL=1,L
      X(1,1)=0.5*X(1,1)+0.5*X(1,2)
      X(2,1)=0.0
      X(1,LF)=0.5*X(1,LF)+0.5*X(1,LF-1)
      X(2,LF)=0.0
      IND=2
      DO 2 JL=3,LFM1,2
      X(2,JL)=0.25*X(2,JL-1)+0.5*X(2,JL)+0.25*X(2,JL+1)
      X(1,JL)=0.25*X(1,JL-1)+0.5*X(1,JL)+0.25*X(1,JL+1)
      X(1,IND)=X(1,JL)
      X(2,IND)=X(2,JL)
      IND=IND+1
    2 CONTINUE
      X(1,IND)=X(1,LF)
      X(2,IND)=X(2,LF)
      LF=LF/2+1
      LFM1=LF-1
    1 CONTINUE
      RETURN
      END



      SUBROUTINE DOTEM(X,Y,L,Z)
      DIMENSION X(2,L),Y(2,L),Z(2,L)
      DO 1 IL=1,L
      SAVER=X(1,IL)*Y(1,IL)+X(2,IL)*Y(2,IL)
      SAVEI=X(1,IL)*Y(2,IL)-X(2,IL)*Y(1,IL)
      Z(1,IL)=SAVER
    1 Z(2,IL)=SAVEI
      RETURN
      END
```

```
      SUBROUTINE DISC63(IBLOCK,ISWITCH,X,N)
      DIMENSION X(N)
C
C         THIS IS THE SDL DISC DRIVER ROUTINE WRITTEN IN CODAP-1
C         IT TRANSFERS WORDS BETWEEN CORE AND THE DISC
C         IBLOCK IS THE DISC BLOCK (32 WORDS) ADDRESS
C         ISWITCH CONTROLS READING AND WRITING
C              ISWITCH=0   GIVES A READ FROM THE DISC
C              ISWITCH=1 GIVES A WRITE ON THE DISC
C         X IS THE CORE ADDRESS
C         N IS THE NUMBER OF WORDS TO TRANSFER
C
C***********************************************************************
C
C         THIS ROUTINE MUST BE SUPPLIED BY THE USER OR INCLUDED IN BINARY
C
C***********************************************************************
      RETURN
      END
```

```
      SUBROUTINE ERASE(N,X)
      DIMENSION X(N)
C
C         ERASE N WORDS IN X
C
      DO 1 I=1,N
    1 X(I)=0.0
      RETURN
      END
```

```
      SUBROUTINE SKIPREC(N,ITAPE)
C
C         SKIP N LOGICAL RECORDS ON TAPE ITAPE
C
      DO 1 I=1,N
    1 READ(ITAPE)LOST
      RETURN
      END
```

APPENDIX C - PROGRAM WRITE-UPS

FINITE FOURIER TRANSFORM THEORY AND ITS APPLICATION TO THE COMPUTATION OF CONVOLUTIONS, CORRELATIONS, AND SPECTRA

SEISMIC DATA LABORATORY
ALEXANDRIA, VIRGINIA

DIGITAL COMPUTING SECTION

A. IDENTIFICATION

Title: Hyper-Rapid Specialized Cooley-Tukey Fourier Transform (direct only)

COOP Identification: G612-COOL

Category: Fourier Transform

Programers: J. F. Claerbout, D. W. McCowan, J. L. Gibson, and E. A. Flinn

Date: 26 February 1966

B. PURPOSE

To compute the Fourier series expansion of a real-or complex-valued data series, or the data series from the complex-valued Fourier series expansion.

C. USAGE

1. Operational Procedure and Parameters:

This is a CODAP subroutine with a FORTRAN-63 calling sequence CALL COOL (N, X, SIGN). X is a complex array used for the data series and the transform; the number of elements of X is $L = 2^N$; SIGN = -1.0 for a direct Fourier transform, and +1.0 for an inverse Fourier transform (but see below for arrangement of data).

For the direct transform: on input the real part

of X contains the data series and the imaginary part of X is zero. On return, the Fourier cosine series expansion is in the real part of X, and the Fourier sine series expansion is in the imaginary part of X. Each contians only $2^{N-1} + 1$ nonredundant points; the cosine expansion is symmetric about point number $2^{N-1} + 1$ and the sine transform is antisymmetric about this point.

For example: N = 3 and data = (0., 1., 0., 0., 0., 0., 0., 0.); Re(X) = (0., 1., 0., 0., 0., 0., 0., 0.); Im(x) = (0., 0., 0., 0., 0., 0., 0., 0.) On imput. On return, Re(X) = (1.000, .7071, 0., -.7071, -1.000, -.7071, 0., .7071); Im(X) = (0., -.7071, -1.000, -.7071, 0., .7071, 1.000, .7071). Point number 1 corresponds to zero frequency; point number 5 corresponds to π.

For inverse transform: the cosine and sine series must be folded over about point number $2^{N-1} + 1$ before calling COOL with SIGN = +1.0.

There is a scale factor of 2^{-N} which COOL does <u>not</u> apply. The user can choose to apply the scale factor either to the direct or to the inverse transform, or to apply a factor of $2^{-N/2}$ to both. For example, if COOL were called with the transform example above, the result would be Re(X) = (0., 8., 0., 0., 0., 0., 0.) and Im(X) = (0., 0., 0., 0., 0., 0., 0., 0.).

2. Space Required: Approximately 200_{10} exclusive of X. The largest series that can be transformed in a 32K core machine is 8K.

3. Temporary Storage Required: None. Other versions of this program have an auxiliary storage for the cosine table and/or a table of bit-reversed numbers. COOL computes its sines and cosines as it goes, and uses an algorithm due to J. F. Claerbout for calculating the bit-reversed numbers.

4. Printout: None.

5. Error Printouts: None.

6. Error Stops: None.

7. Input and Output Tape Mountings: Not Applicable

8. Input and Output Formats: Not Applicable.

9. Selective Jumps and Stops: None.

10. Timing: Time is proportional to $N \cdot 2^N$. Transforming 8192 on the CDC 1604-B requires 25.0 seconds.

11. Accuracy: Calling COOL returns the original to about nine decimal places.

12. Cautions to User: See Operational Procedure above.

13. Configuration: Standard COOP.

14. References: J. W. Cooley, 1964 "Harm - Harmonic Analysis; Calculation of Complex Fourier Series": IBM Watson Research Center Yorktown Height, New York.

J. W. Cocley and J. W. Tukey, 1965, An Algorithm for the Machine Calculation of Complex Fourier Series: Math. of Comp., Vol. 19, pp. 297-301.

Writeups of the following SDL programs:

COOLTWO: Does two Fourier transforms at once.

FT3DCOOL: Three-dimensional Fourier transform

D. METHOD

Given a time series X(I), 1, L (where $L = 2^N$) assumed to be periodic outside the given range, COOL constructs

$$Y(K) = \sum_{J=0}^{N-1} X(J)*W^{JK} \qquad K = 0, \quad L - 1$$

where $W = \exp(-2\pi i/L)$ for time-frequency transform, and $W = \exp(+2\pi i/L)$ for frequency-time transform. The algorithm is efficient, requiring $N \cdot 2^N$ multiplications rather than 2^{2N}.

SEISMIC DATA LABORATORY
ALEXANDRIA, VIRGINIA

DIGITAL COMPUTING SECTION

A. IDENTIFICATION

Title: Multichannel convolution in the frequency domain, for taped data.

COOP Identification: UES G620 COOLCON

Category: G6 Time Series Analysis

Programer: D. W. McCowan

Date: 22 September 1966

B. PURPOSE

This subroutine convolves data channels on the input subset tape with a multichannel filter stored in core, working entirely in the frequency domain. The result is written in subset format on another tape.

C. USAGE

1. Operational Procedure: This is a FORTRAN-63 subroutine with calling sequence:

```
CALL COOLCON (INT, IOT, L, F, X).
```

2. Parameters:

INT is the number of the input tape unit.

IOT is the number of the output tape unit.

L is the number of points in the filter (see restriction below).

F is the multichannel filter, dimensioned F(N,L) in the calling program, where N is the number of channels on the input subset tape.

X is a working array, dimensioned X(2,IT) is the calling program, where IT is the least power of 2 such that

$$2^{IT} > L + LX$$

where LX is the number of data points in the input channels.

Restriction on length of data and length of filter:

LX + L must not be greater than 2^{13} (8K).

3. Space Required: Very little in addition to arrays.

4. Temporary Storage Required: $2*2^{IT}$ working space, plus 127_{10} for the subset tape label.

5. Printout: None.

6. Error Printouts: If $L+LX>2^{13}$, these numbers are printed with an error message.

7. Error Stops: If $L+LX>2^{13}$, the subroutine stops the calling program.

8. Input and Output Tape Mountings: See Parameters above.

9. Input and Output Formats: Compatible with UES Subset (See Writeup).

10. Selective Jump and Stop Settings: None.

11. Timing: Dominated by two Fourier transforms using COOL

for each channel to be filtered. The length of transform is 2^{IT} (See Writeup of COOL).

12. Accuracy: This yields the same numbers, to ten decimal places, which would be computed by convolving the filter and data series in the usual way.

13. Cautions to User: None.

14. Configuration: Standard COOP.

15. References: Writeups of UES G612 COOL, UES Z24 SUBSET, and UES G617 COOLER.

D. METHOD

For each channel to be filtered, the subroutine erases 2^{IT+1} locations of X, and multiplexes the filter and the data channel in X, starting at the beginning. Note that as far as COOL is concerned, X is a complex array with data in the real part and filter in the imaginary part. COOL is called, and the logic of COOLER (q.v.) is used to form the Fourier transform of the filtered channel in X. COOL is called again to get back to the time domain, and the filtered channel is written on the output tape.

The subset label is copied from the input tape to the output tape at the beginning of the subroutine.

SEISMIC DATA LABORATORY
ALEXANDRIA, VIRGINIA

DIGITAL COMPUTING SECTION

A. IDENTIFICATION

Title: Hyper-Rapid Specialized Cooley-Tukey Fourier Transform (direct only)

COOP Identification: G617-COOLER

Category: Fourier Transform

Programer: J. F. Claerbout

Date: 27 July 1966

B. PURPOSE

To compute the Fourier series expansion of a real-valued time series.

C. USAGE

1. Operational Procedure: This is a FORTRAN-63 subroutine, with calling sequence CALL COOLER(N,X). This subroutine calls COOL.

2. Parameters: On input, X is a real-valued time series containing LX points, where $LX = 2^N$, N is restricted to be 14 or less. On return, X contains ½LX+1 complex points of the Fourier transform of the data, with the real and imaginary parts multiplexed together - i. e., on return X can be thought of as a complex array, with the cosine transform in the real part and the sine

transform in the imaginary part.

X must be dimensioned at least LX+2 in the calling program. (i.e., ½LX+1 complex points)

3. Space Required: Very little.

4. Temporary Storage Required: None.

5. Printout: None.

6. Error Printouts: None.

7. Error Stops: None.

8. Input and Output Tape Mountings: Not Applicable.

9. Input and Output Formats: None.

10. Selective Jumps and Stops: None.

11. Timing: Time is proportional to $N.2^N$; transforming 16384 points on the CDC 1604-B requires 45.0 seconds.

12. Accuracy: About nine decimal places.

13. Cautions to User: On return, the real and imaginary parts of the transform are multiplexed together. X must be dimensioned at least LX+2 in the calling program, not LX. This subroutine will not do an inverse transform.

14. References: Writeup of UES G612 COOL.

SEISMIC DATA LABORATORY
ALEXANDRIA, VIRGINIA

DIGITAL COMPUTING SECTION

A. IDENTIFICATION

Title: Hilbert transform of periodic data

COOP Identification: UES G619 COOLHLBR

Category: G6 Time Series Analysis

Programer: E. A. Flinn and J. F. Claerbout

Date: 23 September 1966

B. PURPOSE

To compute the Hilbert transform (quadrature function) of a time series. Since COOL is used, the time series is assumed to be periodic outside the range of definition.

C. USAGE

1. Operational Procedure: This is a FORTRAN-63 subroutine, with calling sequence: CALL COOLHLBR(N,X). This subroutine calls COOL.

2. Parameters: N is the log (base 2) of the number of data points. X is the data, dimensioned at least 2^N in the calling program, and type complex there.

On input, the real data series must be stored in the real part of X, and the imaginary part must be zero.

On return, the real data series is stored in the real part of scaled up by 2^{N-1}. The Hilbert transform is stored in the imaginary part of X, also scaled up by 2^{N-1}.

3. <u>Space Required</u>: Very little in addition to the array for data, which requires 2^{N+1} locations in the calling program.

4. <u>Temporary Storage Required</u>: None

5. <u>Printout</u>: None

6. <u>Error Printouts</u>: None

7. <u>Error Stops</u>: None

8. <u>Input and Output Tape Mountings</u>: Not Applicable

9. <u>Input and Output Formats</u>: Not Applicable

10. <u>Selective Jumps and Stops</u>: None

11. <u>Timing</u>: Dominated by two calls to COOL

12. <u>Accuracy</u>: The data is returned correct to ten decimal places.

13. <u>Cautions to User</u>: The data must be arranged as under (2) above.

    Notice that as far as this subroutine is concerned the data is periodic outside the range of definition. End effects may cause answers which the user does not expect. For example, if the input is a pure sine wave, the user expects the quadrature to be a pure cosine. Using this subroutine, this turns out to be the case only if the data series contains an integral number of cycles.

14. <u>References:</u> Writeup of UES G612 COOL.

D. METHOD

The Hilbert transform of a function has a Fourier transform which is $(-1)^{\frac{1}{2}}$ times the Fourier transform of the function. COOL returns the real and imaginary parts of the Fourier transform of a function calculated from zero to 2π, so that the real part is symmetric about the middle and the imaginary part is antisymmetric.

If the Fourier transform of the function is A+iB, the Fourier transform of the Hilbert transform is -B+iA. All COOLHLBR does is erase the second half of the Fourier transform (the part from π to 2π), half-weight the end points, and call COOL again to transform back to the time domain.

The scale factor 2^{N-1} comes from the fact that COOL gives the unnormalized transform.

SEISMIC DATA LABORATORY
ALEXANDRIA, VIRGINIA

DIGITAL COMPUTING SECTION

A. IDENTIFICATION

Title: Fourier Transform of Two Data Series Simultaneously

COOP Identification: COOLTWO

Category: G6 Time Series Analysis

Programer: E. A. Flinn

Date: 10 June 1966

B. PURPOSE

To compute the Fourier series expansion, using COOL (q.v.), of two data series simultaneously.

C. USAGE

1. Operational Procedure: This is a FORTRAN-63 subroutine with calling sequence.

CALL COOLTWO (N, X, SIGN, A, B).

2. Parameters:

N is the log (base 2) of the number of elements in X;

X contains the two data series, multiplexed in one complex array, so that Re(X) contains one series and Im(X) contains the other.

SIGN = -1.0 . The program has not yet been checked out for inverse transformation;

A is the complex (cosine and sine) transform of the data series stored in the real part of X;

B is the complex Fourier transform of the data series stored in the imaginary part of X;

A and B are both of length 2**(N - 1) + 1.

3. Space Required: about 70_{10} excluding arrays.

4. Temporary Storage Requirements: None

5. Printouts: None

6. Error Printouts: None

7. Error Stops: None

8. Input and Output Tape Mountings: None

9. Input and Output Formats: Not Applicable

10. Selective Jump and Stop Settings: Not Applicable

11. Timing: Timing is proportional to $N \cdot 2^N$; transforming 8192 data points on the CDC 1604-B requires 25.0 seconds.

12. Accuracy: Same as COOL.

13. Cautions to User: This program has not been checked out for inverse transformation. This program does not apply the scale factor 2^{-N}, since some users may wish to apply the scale factor to the inverse, rather than the direct transform. The number of data points must be a power of 2.

14. Configuration: Standard COOP

15. References: Writeup of UES G612 COOL

D. METHOD

The method is due to J. W. Cooley (see Reference 2 in main body of this report.)

SEISMIC DATA LABORATORY
ALEXANDRIA, VIRGINIA

DIGITAL COMPUTING SECTION

A. IDENTIFICATION

Title: Fast convolution of two time series using COOL.

COOP Identification: UES COOLVOLV

Category: Time Series Analysis

Programer: E. A. Flinn and D. W. McCowan

Date: 23 September 1966

B. PURPOSE

To form the convolution of two time series, not by the usual polynomial multiplication algorithm, but by forming the two Fourier transforms (using COOL), multiplying them together, and transforming back to the time domain. This is faster than the usual procedure when

$$LX \cdot LF \gg 4\ (2N + 1) \quad (LX + LF)$$

where LX is the data series length, LF is the filter impulse response length, and N is the log (base 2) of LX + LF.

C. USAGE

1. Operational Procedure: This is a FORTRAN-63 subroutine, with calling sequence:

CALL COOLVOLV(LX,X,LF,F)

2. Parameters:

X is the data series to be convolved, dimensioned at least

2^{J+1} in the calling program, where 2^J is the smallest power of two larger than LX + LF.

LX is the length of the data series to be convolved.

F is the filter to be convolved with X.

LF is the length of the filter.

3. Space Required: 300_{10} plus arrays.
4. Temporary Locations Required: None beyond filling out X to the first power of two greater the LX + LF.
5. Alarms or Special Printout: None
6. Error Returns: If LX + LF $\geq 2^{13}$, LF is replaced by -LF and control is returned to the calling program.
7. Error Stops: None
8. Tape Mountings: None
9. Formats: None
10. Jump and Stop Settings: None
11. Timing: Dominated by two calls to COOL for LX + LF points each time.
12. Accuracy: Gives the same results as polynomial multiplication to ten decimal places.
13. Cautions: None
14. Configuration: Standard COCP
15. References: Writeups of COOL, COOLCON, AND COOLER

D. METHOD

The same method is used as used in COOLCON.

SEISMIC DATA LABORATORY
ALEXANDRIA, VIRGINIA

DIGITAL COMPUTING SECTION

A. IDENTIFICATION

Title: Two and Three Dimensional Fourier Transform Package

COOP Identification: G615 FT2DCOOL, FT3DCOOL

Category: G6 Time Series Analysis

Programer: D. W. McCowan

Date: 20 April 1966

B. PURPOSE

The subroutines in this package compute two and three dimensional Fourier transforms. Their names are: FT2DCOOL, FT3DCOOL, COOL, MATRA63, and SCALE. As with COOL, the dimensions on the data must be a power of two.

C. USAGE

1. Calling Sequence:

```
CALL FT2DCOOL (X,N,M, SIGNI)
```

and

```
CALL FT3DCOOL (X,N,M,L, SIGNI)
```

2. Arguments:

X, the complex array in which the data is supplied and in which the Fourier transform is returned. If real data is supplied, it must be put into the real part of X and the imaginary part must be erased.

N,M,L, the dimensions of X. Each of these numbers must be a power of two. The number of complex points in the Fourier transform will be N/2 + 1, M/2 + 1, and L/2 +1 in each direction.

SIGNI, a switch determining the type of transform to be performed. SIGNI = -1.0 gives a direct transform (time to frequency), and SIGNI = +1.0 gives the inverse.

3. Space Required: 500 locations.

4. Temporary Storage: None

5. Alarms and Printouts: None

6. Error Returns: None

7. Error Stops: None

8. Tape Mountings: None

9. Formats: None

10. Jumps and Stop Settings: None

11. Time Required: Three-dimensional Fourier transforms require NM + NL + ML one-dimensional Fourier transforms. Two-dimensional Fourier transforms require N + M one-dimensional Fourier transforms. For the timing of one-dimensional Fourier transforms, see References.

12. Accuracy: Same as COOL

13. Cautions to Users: None

14. Equipment Configuration: Standard COOP

15. References: Writeup of UES G612 COOL 3/30/66

D. METHOD

The direct 2 and 3-dimensional Fourier transforms are defined as:

$$A(j_1, j_2) = \frac{1}{\sqrt{NM}} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{M-1} x(k_1, k_2)\, W_1^{-j_1 k_1}\, W_2^{-j_2 k_2}$$

and

$$A(j_1, j_2, j_3) = \frac{1}{\sqrt{NML}} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{M-1} \sum_{k_3=0}^{L-1} X(k_1, k_2, k_3)$$

$$W_1^{-j_1 k_1}\, W_2^{-j_2 k_2}\, W_3^{-j_3 k_3}$$

Where $W_1 = \exp(2\pi i/N)$; $W_2 = \exp(2\pi i/N)$; $W_3 = \exp(2\pi i/L)$

The two-dimensional transform is broken up into N + M one-dimensional transforms and the three-dimensional transform is broken up into L two-dimensional transforms and NM one-dimensional transforms.

SEISMIC DATA LABORATORY
ALEXANDRIA, VIRGINIA

DIGITAL COMPUTING SECTION

A. IDENTIFICATION

Title: Spectral Matrix Estimates

COOP Identification: G618 SPECTRUM

Category: Time Series Analysis

Programer: D. W. McCowan

Date: 10 July 1966

B. PURPOSE

This is a package of three FORTRAN-63 subroutines for computing an estimate of the spectral matrix for N channels of data stored on magnetic tape. It uses the hyper-rapid Fourier transform routine COOL, and makes use of two tapes and the disc to cut running time to a minimum. The names of the three routines in the package are: SPECTRUM, DOTEM, and SMOOTH. In addition to these, three more subroutines are assumed to be on the system tape; they are: COOL, SKIPREC, and ERASE. Since all other routines are called internally by SPECTRUM, only the calling sequence for it will be given.

C. USAGE

1. Calling Sequence:

Call SPECTRUM (IT, JT, KT, S, NS, LF, X)

2. <u>Arguments</u>:

IT, the input subset tape number on which the N channels of data are written. The length of each channel must be exactly a power of two.

JT, the number of a scratch tape.

KT, the number of a scratch tape.

S, a triply subscripted FORTRAN-63 complex array used both for internal manipulation and to return the computed spectral matrix as a N by N by LF complex array with subscripts varying in that order. Here N is the number of channels read from the input tape label and LF is the smoothed length of each spectral estimate. This array must also be 4*LX+4 locations in length, since it is also used for internal computations. LX is the length of the input data channels read from the input tape label. Remembering that there are two locations used for each complex number, the total dimensions on S in the main program must be 2*N*N*LF or 4*LX+4, whichever is the larger. It is usually convenient to dimension it as complex N by N by L F63 array in order to facilitate use. L here is a number chosen so that S will be large enough as described above.

NS, the number of times to apply the hanning smoothing operation to the original estimates.

LF, the returned length of the spectral estimates. This is computed from the formula:

LF = (LX/(2**NS) + 1

LF must not be larger than 129.

X, an array used for internal manipulation, containing at least 2*LF locations.

3. Space Required: 502 locations
4. Temporary Locations: None
5. Alarms or Special Printout: None
6. Error Returns: None
7. Error Stops: The subroutines stop if length of data series exceeds 2^{13}.
8. Tape Mountings: See Arguments
9. Input and Output Formats: See Arguments
10. Jump Settings: None
11. Time Required: A 10-channel, 4096-point, NS = 6 case takes approximately 10 minutes of 1604 time.
12. Accuracy: Single precision
13. Caution to Users: The subroutine as written requires that the data series should contain a number of points exactly a power of two.
14. Equipment Configuration: Standard COOP
15. References: Writeup of subroutine G612COOL, 6/1/66

Writeup of program UES Z24 SUBSET

Stockham, T. G., 1966 High Speed Convolution and Correlation, AFIPS Proceedings

D. METHOD

The spectral matrix elements $S_{ij}(k)$ are usually defined as Fourier transforms of correlation functions $R_{ij}(t)$. However, it must be realized that these correlations are transient correlations where the functions are considered to be zero outside the region of interest and 100% lags are taken. They are defined as follows:

$$R_{ij}(t) = \sum_{\tau=0}^{T-1-t} x_i(\tau)\, x_j(\tau + t)$$

$$R_{ij}(-t) = \sum_{\tau=t}^{T-1} x_i(\tau)\, x_j(\tau - t) = R_{ji}(t)$$

The spectral matrix element is then

$$S_{ij}(k) = \sum_{t=0}^{T-1} \sum_{\tau=t}^{T-1} x_i(\tau)\, x_j(\tau - t)\, W \frac{tk}{2} + \sum_{t=1}^{T-1} \sum_{\tau=0}^{T-1-t}$$

$$x_i(\tau)\, x_j(\tau+t)\, W \frac{-tk}{2}$$

This can be shown to be equivalent to:

$$S_{ij}(k) = F_i^*(k)\, F_j(k).$$

where

$$F_i(k) = \sum_{t=0}^{T-1} x_i(t)\, W^{-\frac{tk}{2}}$$

This is recognized as the Fourier transform of the input data computed over twice its length with zeros filled into the second half. The Cooley-Tukey hyper-rapid Fourier transform routine COOL is used to provide the high speed necessary here.

Each spectral matrix element is originally T + 1 complex points long between DC and the folding frequency. It is then smoothed with a hanning window NS times to its final length of LF points.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R&D	
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)	
1. ORIGINATING ACTIVITY (Corporate author) TELEDYNE, INC. ALEXANDRIA, VIRGINIA	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP ----

3. REPORT TITLE
FINITE FOURIER TRANSFORM THEORY AND ITS APPLICATION TO THE COMPUTATION OF CONVOLUTIONS, CORRELATIONS AND SPECTRA
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Scientific
5. AUTHOR(S) (Last name, first name, initial) McCowan, Douglas W.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October 17, 1967	62	6

8a. CONTRACT OR GRANT NO. F 33657-67-C-1313	9a. ORIGINATOR'S REPORT NUMBER(S) 168 (Revised)
b. PROJECT NO. VELA T/6702	
c. ARPA Order No. 624	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) ----
d. ARPA Program Code No. 5810	

10. AVAILABILITY/LIMITATION NOTICES
This document is subject to special export controls & each transmittal to foreign governments or foreign national may be made only with prior approval of Chief, AFTAC.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
----	ADVANCED RESEARCH PROJECTS AGENCY NUCLEAR TEST DETECTION OFFICE WASHINGTON, D. C.

13. ABSTRACT

The theory of finite Fourier transforms is developed from the definitions of infinite transforms and applied to the computation of convolutions, correlations, and power spectra. Detailed procedures for these computations are given, including listings & writeups of FORTRAN subroutines.

DD FORM 1 JAN 64 1473

Unclassified
Security Classification

Unclassified

Security Classification

14. KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
Fourier Transforms						
Seismic Array Data						
Digital Computer Data Processing						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through ______."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through ______."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through ______."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

Unclassified

Security Classification